Co-funded by the
Erasmus+ Programme
of the European Union

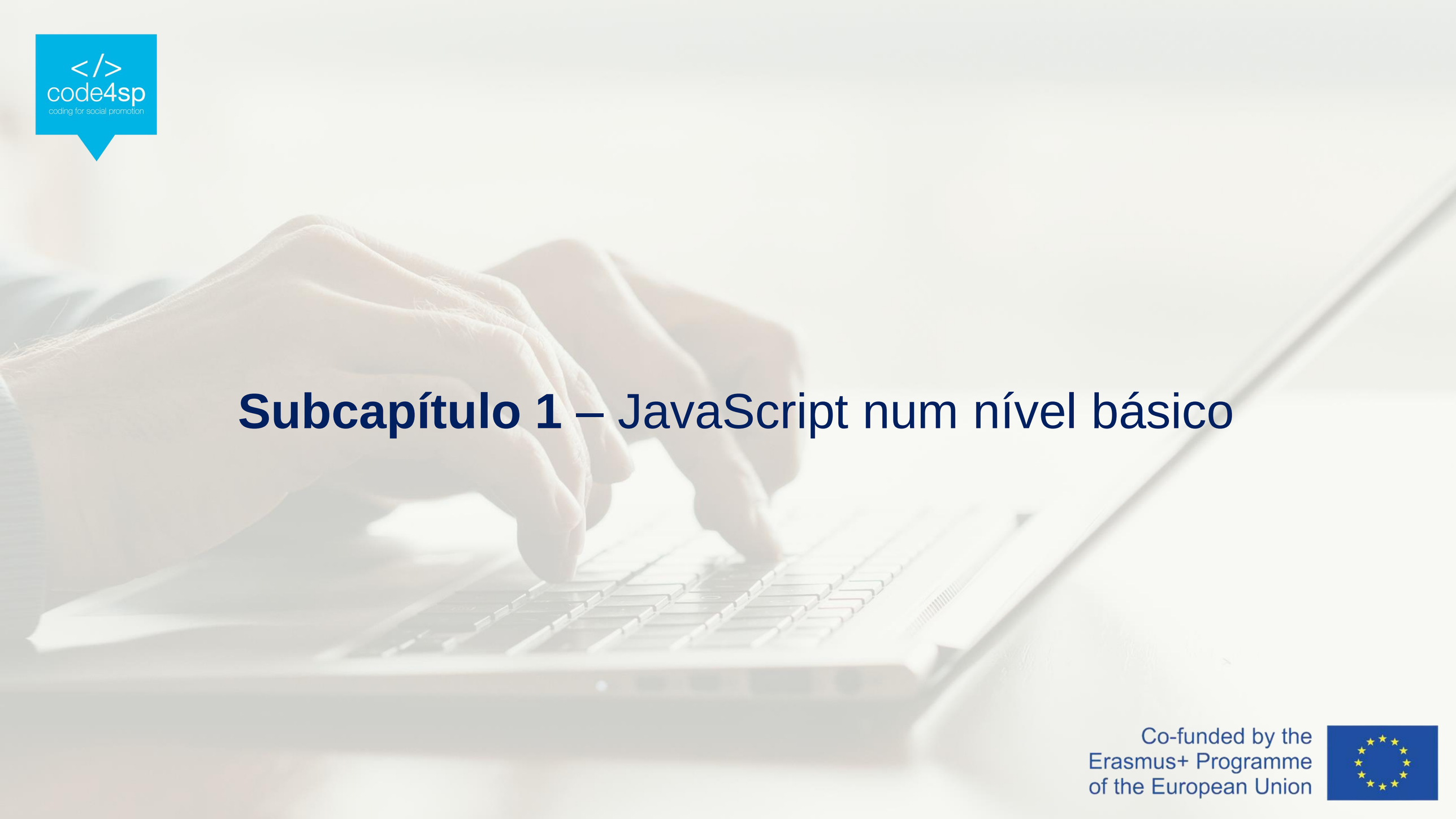
code4sp
coding for social promotion

Materiais de Formação em JavaScript
Subcapítulo 1 – JavaScript num nível básico

WP3: Materiais de Formação do Code4SP

Elaborado por:

The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein. (Project Number: 621417-EPP-1-2020-1-PT-EPPKA3-IPI-SOC-IN)



Subcapítulo 1 – JavaScript num nível básico

Co-funded by the
Erasmus+ Programme
of the European Union

O que é o JavaScript?

- JavaScript (JS) é a linguagem mais abrangente no scripting da visão do cliente (o scripting da visão do cliente está associada com a execução num web browser).JS pretende adicionar, às páginas web, efeitos interativos e dinâmicos através da manipulação do conteúdo devolvido de um servidor web.

- O JavaScript é uma linguagem orientada do objeto, tendo também algumas similaridades na sintaxe da linguagem de programação Java, mesmo que não seja relacionado, de todo, ao Java.

O JavaScript pode ser usado para vários fins:

- Modificar o conteúdo de uma página web ao adicionar ou remover elementos;
- Alterar o estilo e posição dos elementos numa página web;
- Monitorizar eventos como o clicar do rato, hover, etc. e reagir a isso;
- Fazer e controlar transições na página web;
- Produzir alertas pop-ups para mostrar mensagens de informação e aviso para o utilizado;
- Completar operações baseadas no contributo do utilizador e na exibição dos resultados;
- Validar o contributo do utilizador antes de submete-los no servidor;
- E muitos outros propósitos interessantes que serão mostrados mais tarde

Co-funded by the
Erasmus+ Programme
of the European Union

Iniciar com o JavaScript

Neste ponto, os alunos irão entender o quão simples pode ser adicionar interatividade numa página web ao usar o JavaScript.

Caracteristicamente, existem 3 formas de adicionar JS a uma página web:

a) **Incorporar o código JavaScript entre a tag <script> e </script>;**

b) Criar um ficheiro JavaScript com a extensão .js e depois abrir dentro da página através do atributo da tag <script>.

c) Colocar o código JavaScript diretamente dentro da tag HTML usando a tag especial de atributo como onclick, onmouseover, onkeypress, onload, etc.

Co-funded by the
Erasmus+ Programme
of the European Union

Iniciar com o JavaScript

a) Incorporar o código JavaScript entre a tag <script> e </script>

O código JavaScript pode ser incorporado diretamente no interior de uma página web ao colocá-lo entre as tags <script> e </script>. A tag <script> indica ao browser que as afirmações presentes devem ser interpretadas como um guião exequível e não como HTML, tal como mostrado no seguinte exemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Embedding JavaScript</title>
</head>
<body>
    <script>
    var greet = "Hello World!";
    document.write(greet); // Prints: Hello World!
    </script>
</body>
</html>
```

Hello World!

Co-funded by the
Erasmus+ Programme
of the European Union

Iniciar com o JavaScript

b) Criar um ficheiro JavaScript com a extensão .js e depois abrir dentro da página através do atributo da tag <script>.

- O código JavaScript pode ser colocado, também, em ficheiros separados com a extensão .js, sendo então chamado para o ficheiro no mesmo documento através do atributo src da tag <script>, assim:

```
<script src="js/hello.js"></script>
```

- Esta é especialmente valiosa se o programador quiser que os mesmos scripts estejam disponíveis em vários documentos. Ao seguir este procedimento, ele/ela vai evitar repetir a mesma tarefa vezes e vezes sem conta, e faz com que o seu website seja mais fácil de manter.

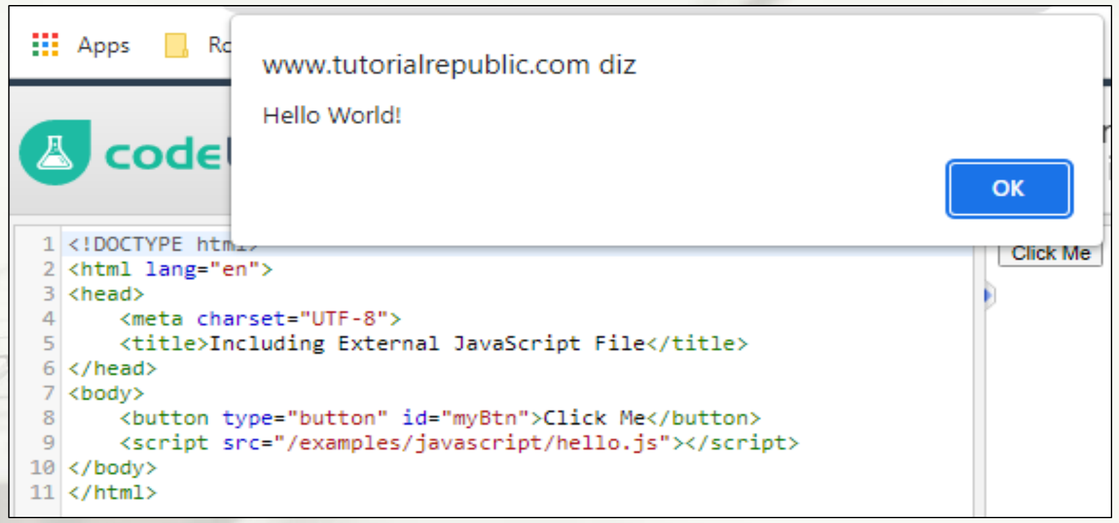
Co-funded by the Erasmus+ Programme of the European Union

Iniciar com o JavaScript

Seguindo para a afirmação em cima, será criado um ficheiro JavaScript denominado como “**hello.js**” e será introduzido o código seguinte.

```javascript
// A function to display a message
function sayHello() {
    alert("Hello World!");
}

// Call function on click of the button
document.getElementById("myBtn").onclick = sayHello;
```



Co-funded by the
Erasmus+ Programme
of the European Union

Iniciar com o JavaScript

c) Colocar o código JavaScript diretamente dentro da tag HTML usando a tag especial de atributo como onclick, onmouseover, onkeypress, onload, etc.

O código JavaScript pode ser introduzido inline ao inseri-lo diretamente dentro da tag HTML através dos atributos especiais como onclick, onmouseover, onkeypress, onload, etc.

Co-funded by the
Erasmus+ Programme
of the European Union

Iniciar com o JavaScript

Independentemente, não é aconselhado colocar grandes quantidades de código JavaScript inline pois pode desordenar o HTML com o JavaScript e fazer com que o código JS possa ser mais difícil de manter. A figura 4 mostra um exemplo (neste caso, uma mensagem de alerta é mostrada ao clicar no elemento de botão):

Co-funded by the
Erasmus+ Programme
of the European Union

Iniciar com o JavaScript

- O elemento <script> pode ser posicionado na secção <head> ou <body> de um documento HTML. No entanto, os scripts devem ser preferencialmente posicionados no fim da secção do corpo, antes de fechar a tag </body>.

- Este procedimento permite às páginas web carregar mais rapidamente, pois evita obstruções que possam acontecer na renderização inicial da página. Cada tag <script> bloqueia o processo de renderização até este estar completamente descarregado e executado pelo código JavaScript, colocando-os, então, na secção de head (ex. o elemento <head>) do documento sem qualquer razão válida que irá impactar, significativamente, o desempenho do website.

Co-funded by the
Erasmus+ Programme
of the European Union

Iniciar com o JavaScript

- **Existem diferenças entre o scripting da visão de cliente e a visão de servidor.**

- A linguagem de visão do cliente (p.e., o JavaScript ou VBScript são interpretados e executados pelo browser da web, ao contrário do scripting da linguagem da visão do servidor (p.e.; PHP, ASP, Java, Python, Ruby, etc.), o que corre num servidor web e o seu contributo é enviado de volta para o browser de web em formato HTML.

- O scripting da visão do cliente tem várias vantagens comparadas com o scripting tradicional da visão do sevidor. Por exemplo, o JavaScritpt pode ser usado para verificar se o utilizador inseriu dados inválidos em campos de formulário e mostrar a notificação, em tempo real, dos erros cometidos antes de submeter o formulário ao servidor web para a validação final dos dados e o processamento seguinte, de forma a evitar usos desnecessários de bandwidth de networks e do uso incorreto de recursos de sistemas de servidores.

Co-funded by the
Erasmus+ Programme
of the European Union

Síntaxe do JavaScript

- A sintaxe do JS é um **conjunto de regras** que compilam uma programação bem estruturada em JavaScript. JS envolve afirmações que são colocadas entre as tags HTML <script> e </script> numa página web, ou dentro do ficheiro externo de JavaScript com a extensão .js .

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Example of JavaScript Statements</title>
</head>
<body>
    <script>
    var x = 5;
    var y = 10;
    var sum = x + y;
    document.write(sum); // Prints variable value
    </script>
</body>
</html>
```

```
15
```

Co-funded by the
Erasmus+ Programme
of the European Union

Síntaxe do JavaScript

- Os formandos devem notar que o JavaScript é sensível a maiúsculas e minúsculas. Desta forma, as variáveis, palavras-chave de linguagem, nomes de funções e outros identificadores devem ser escritos sempre iguais. Desta forma, a variável myVar deve ser sempre escrita desta forma (e não, “MYVAR”, “myvar”, etc.). **Isto é aplicável em todos os casos**.

- De acordo com os tópicos anteriores, o JavaScript permite escrever comentários ao longo das linhas de código. Os comentários são inseridos maioritariamente porque dão informação extra à fonte de código, mas também porque pode ajudar programadores a entender os seus códigos após algum tempo, trabalho em equipa, etc.

Co-funded by the
Erasmus+ Programme
of the European Union

Síntaxe do JavaScript

É possível adicionar tanto linhas únicas como várias linhas de comentários no JavaScript. Os comentários de linha única começam como uma dupla barra para a frente (//), seguindo-se o texto com o comentário:

```
1  // This is my first JavaScript program
2  document.write("Hello World!");
```

Co-funded by the
Erasmus+ Programme
of the European Union

Síntaxe do JavaScript

Para um comentário com mais de uma linha, o ponto de começo são uma barra e um asterisco (/*), acabando com um asterisco e uma barra (*/):

```
1  /* This is my first program
2  in JavaScript */
3  document.write("Hello World!");
```

Co-funded by the
Erasmus+ Programme
of the European Union

Variáveis em JavaScript

- Para armazenar dados em JavaScript, os programadores criam variáveis.

- Estas são a chave de todas as linguagens de programação e são usadas para armazenar dados, por exemplo, uma linha de texto, números ou outros elementos.

- Quando necessitar, o programador pode determinar, atualizar ou recuperar dados ou valores guardados nas variáveis. As variáveis podem ser entendidas como nomes simbólicos para os valores.

Co-funded by the
Erasmus+ Programme
of the European Union

Variáveis em JavaScript

- Uma variável pode ser criada usando a palavra-chave var, onde o operador designado (“=”) é usado para atribuir um valor à variável, como no seguinte exemplo: *var varName = value*

```
1  var name = "Peter Parker";
2  var age = 21;
3  var isMarried = false;
```

Foram criadas 3 variáveis

Co-funded by the
Erasmus+ Programme
of the European Union

Variáveis em JavaScript

- A última revisão de JavaScriot (ECMAScript 2015 ou ES6) introduziu duas novas palavras-chave para declarar variáveis: let e const.
- A palavra-chave const funciona da mesma forma que a let. No entanto, as variáveis declaradas com “const” não pode ser reatribuídas mais tarde no código, tal como no exemplo:

```
1  // Declaring variables
2  let name = "Harry Potter";
3  let age = 11;
4  let isStudent = true;
5
6  // Declaring constant
7  const PI = 3.14;
8  console.log(PI); // 3.14
9
10 // Trying to reassign
11 PI = 10; // error
```

Contrariamente à palavra-chave “var”, que declara a variável function-scoped, tanto “let” como “const” são palavras-chave que declaram variáveis, delimitando-as a um nível de bloco ({}). A delimitação por bloco significa que uma nova delimitação é criada dentro de um par de chavetas.

Co-funded by the
Erasmus+ Programme
of the European Union

Variáveis em JavaScript

As variáveis JavaScript têm regras específicas para serem nomeadas:

• O nome de uma variável deve começar com uma letra, underscore (_), ou sinal de dólar ($).

• O nome de uma variável não pode começar com um número.

• O nome de uma variável só pode ter caracteres alfa-numéricos (A-Z, 0-9) e underscores.

• O nome de uma variável não pode conter espaços

• O nome de uma variável não pode ser uma palavra-chave do JavaScript ou uma palavra reservada de JavaScript.

Co-funded by the
Erasmus+ Programme
of the European Union

Gerar outputs em JavaScript

- Existem certas situações em que os programadores podem precisar de criar outputs (“resultados”) do código JS, por exemplo, verificar qual o valor de uma variável, escrever uma mensagem no controlo de browser, etc. Em Java Script, existem algumas maneiras de criar outputs, incluindo a sua inscrição na janela do browser ou no controlo do browser, mostrando resultados em caixas de diálogo, escrevendo os resultados num elemento HTML, etc.

- Não é difícil criar *outputs* de uma mensagem ou escrever os dados no controlo do browser (pode ser acedido ao carregar na tecla F12). Para tal, deve ser aplicado o console.log () – um método simples, contudo, poderoso.

- Para escrever o conteúdo do documento corrente enquanto o documento está a ser construído, é necessário desconstruí-lo. O método document. write () pode ser usado.

Co-funded by the
Erasmus+ Programme
of the European Union

Gerar outputs em JavaScript

- Se o método document.write estiver a ser usado depois da página ter carregado, irá reescrever todo o conteúdo existente naquele documento, como explicado neste link..

- **Caixas de diálogos de alerta** podem também ser adicionados para mostrar os dados de mensagem ou resultado ao utilizador. Para criar este diálogo de alerta, o método alert() é usado, tal como no seguinte exemplo:

```
1  // Displaying a simple text message
2  alert("Hello World!"); // Outputs: Hello World!
3
4  // Displaying a variable value
5  var x = 10;
6  var y = 20;
7  var sum = x + y;
8  alert(sum); // Outputs: 30
```

by the
ramme
of the European Union

Gerar outputs em JavaScript

- Os resultados podem ser inseridos ou digitados dentro de um elementoo HTML usando a propriedade innerHTML. Contudo, o programador deve selecionar o elemento antes de digitar o *output,* usando o método getElementById().

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Writing into an HTML Element with JavaScript</title>
</head>
<body>
    <p id="greet"></p>
    <p id="result"></p>

    <script>
    // Writing text string inside an element
    document.getElementById("greet").innerHTML = "Hello World!";

    // Writing a variable value inside an element
    var x = 10;
    var y = 20;
    var sum = x + y;
    document.getElementById("result").innerHTML = sum;
    </script>
</body>
</html>
```

Hello World!

30

Co-funded by the
Erasmus+ Programme
of the European Union

Tipos de Dados em JavaScript

Os tipos de dados, essencialmente, estipulam qual é o tipo de dados que podem ser guardados e manipulados dentro de um programa. Existem seis tipos básicos de dados em JS, os quais podem ser divididos em três categorias principais:

- **Primitivo (ou primário) –** Linha, Número e Boolean são exemplos de tipos de dados primitivos, que apenas conseguem representar um valor de cada vez;
- **Composto (ou referência) –** Objeto, Matriz e Função (que são tipos de objetos) são tipos de dados compostos. Estes podem representar coleções de valores e de entidades mais complexas; e
- **Tipos de dados especiais –** Indefinidos e vazios são tipos de dados especiais.

Co-funded by the
Erasmus+ Programme
of the European Union

Tipos de Dados em JavaScript

O tipo de dados “string”

É usado para incorporar dados textuais (por exemplo, uma sequência de caracteres).

As linhas são criadas usando aspas uma ou duas vezes rodeando um ou mais caracteres:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript String Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var a = 'Hi there!';  // using single quotes
    var b = "Hi there!";  // using double quotes

    // Printing variable values
    document.write(a + "<br>");
    document.write(b);
    </script>
</body>
</html>
```

```
Hi there!
Hi there!
```

Co-funded by the Erasmus+ Programme of the European Union

Tipos de Dados em JavaScript

O tipo de dados “number”

O tipo de dados de números é útil aquando da exibição de números positivou ou negativos com, ou sem, casas decimais, ou números escritos usando uma anotação exponencial, por exemplo: 1.5e-4 (equivalente a $1.5x10^{-4}$)

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <title>JavaScript Number Data Type</title>
6  </head>
7  <body>
8      <script>
9      // Creating variables
10     var a = 25;
11     var b = 80.5;
12     var c = 4.25e+6;
13     var d = 4.25e-6;
14
15     // Printing variable values
16     document.write(a + "<br>");
17     document.write(b + "<br>");
18     document.write(c + "<br>");
19     document.write(d);
20     </script>
21 </body>
22 </html>
```

```
25
80.5
4250000
0.00000425
```

Co-funded by the
Erasmus+ Programme
of the European Union

Tipos de Dados em JavaScript

O tipo de dados “number”

O tipo de dados de número comprime, também, alguns valores especiais: infinito, -infinito e NaN. O infinito representa o infinito matemático (∞), que é maior que qualquer número. O infinito é o resultado da divisão de um número maior que 0, por 0, como pode ser verificado na imagem abaixo:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Infinity</title>
</head>
<body>
    <script>
    document.write(16 / 0);
    document.write("<br>");
    document.write(-16 / 0);
    document.write("<br>");
    document.write(16 / -0);
    </script>
</body>
</html>
```

```
Infinity
-Infinity
-Infinity
```

Co-funded by the Erasmus+ Programme of the European Union

Tipos de Dados em JavaScript

O tipo de dados booleano

Existem dois valores que podem ser integrados neste tipo de dados: verdadeiro ou falso. É, por norma, utilizado para valores de stock como sim (verdadeiro) ou não (falso), ligado (verdadeiro) ou desligado (falso), etc., tal como mostra o exemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Boolean Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var isReading = true;   // yes, I'm reading
    var isSleeping = false; // no, I'm not sleeping

    // Printing variable values
    document.write(isReading + "<br>");
    document.write(isSleeping);
    </script>
</body>
</html>
```

```
true
false
```

Co-funded by the
Erasmus+ Programme
of the European Union

Tipos de Dados em JavaScript

O tipo de dados indefinido

O tipo de dados indefinidos pode, apenas, representar um valor – o valor especial indefinido. Se uma variável for declarada, mas não lhe tiver sido atribuída um valor, o valor deverá ser declarado como indefinido.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Undefined Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var a;
    var b = "Hello World!"

    // Printing variable values
    document.write(a + "<br>");
    document.write(b);
    </script>
</body>
</html>
```

```
undefined
Hello World!
```

Co-funded by the Erasmus+ Programme of the European Union

Tipos de Dados em JavaScript

O tipo de dados nulo

Este é um tipo de dados especial que apenas pode ter um único valor – o valor nulo. Um valor nulo significa que não existe um valor. Não é equivalente a uma linha vazia (“”) ou zero, é simplesmente, nada. Esta variável pode ter os seus conteúdos esvaziados através da atribuição de um valor nulo.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Null Data Type</title>
</head>
<body>
    <script>
    var a = null;
    document.write(a + "<br>"); // Print: null

    var b = "Hello World!"
    document.write(b + "<br>"); // Print: Hello World!

    b = null;
    document.write(b) // Print: null
    </script>
</body>
</html>
```

```
null
Hello World!
null
```

Co-funded by the
Erasmus+ Programme
of the European Union

Tipos de Dados em JavaScript

O tipo de dados de objeto

O objeto é um tipo de dados multifacetado que permite armazenar coleções de dados.

Um objeto contem propriedades definidas por um par de valores chave. Uma chave propriedade (nome) é sempre uma linha, mas o seu valor pode ser qualquer tipo de dados (linhas, números, booleanos, ou tipos de dados complexos, como matrizes, funções e outros objetos). A forma mais simples de criar um objeto em JavaScript é mostrada de seguida:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Object Data Type</title>
</head>
<body>
    <script>
    var emptyObject = {};
    var person = {"name": "Clark", "surname": "Kent", "age": "36"};

    // For better reading
    var car = {
        "modal": "BMW X3",
        "color": "white",
        "doors": 5
    }

    // Print variables values in browser's console
    console.log(person);
    console.log(car);
    </script>
    <p><strong>Note:</strong> Check out the browser console by pressing the f12 key on the keyboard.</p>
</body>
</html>
```

Note: Check out the browser console by pressing the f12 key on the keyboard.

Co-funded by the Erasmus+ Programme of the European Union

Tipos de Dados em JavaScript

The Function Data Type

A função é um objeto que faz um bloco de código. As funções são objetos, daí ser possível designar-lhes variáveis, assim:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Function Data Type</title>
</head>
<body>
    <script>
    var greeting = function(){
        return "Hello World!";
    }

    // Check the type of greeting variable
    document.write(typeof greeting) // Output: function
    document.write("<br>");
    document.write(greeting());     // Output: Hello World!
    </script>
</body>
</html>
```

```
function
Hello World!
```

Co-funded by the
Erasmus+ Programme
of the European Union

Tipos de Dados em JavaScript

O Operador Typeof

O operador typeof pode ser usado para perceber o tipo de dados cobertos por uma variável. Pode ser usado com ou sem parênteses (typeof(x) ou typeof x). O operador typeof é, na sua maioria, benéfico no processo de valorização de diferentes tipos numa forma diferente. No entanto, o programador deve ser cauteloso, pois pode produzir, em alguns casos, resultados imprevistos:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript typeof Operator</title>
</head>
<body>
    <script>
    // Numbers
    document.write(typeof 15 + "<br>");  // Prints: "number"
    document.write(typeof 42.7 + "<br>");  // Prints: "number"
    document.write(typeof 2.5e-4 + "<br>");  // Prints: "number"
    document.write(typeof Infinity + "<br>");  // Prints: "number"
    document.write(typeof NaN + "<br>");  // Prints: "number". Despite being "Not-A-Number"

    // Strings
    document.write(typeof '' + "<br>");  // Prints: "string"
    document.write(typeof 'hello' + "<br>");  // Prints: "string"
    document.write(typeof '12' + "<br>");  // Prints: "string". Number within quotes is document.write(typeof string

    // Booleans
    document.write(typeof true + "<br>");  // Prints: "boolean"
    document.write(typeof false + "<br>");  // Prints: "boolean"

    // Undefined
    document.write(typeof undefined + "<br>");  // Prints: "undefined"
    document.write(typeof undeclaredVariable + "<br>"); // Prints: "undefined"

    // Null
    document.write(typeof Null + "<br>");  // Prints: "object"

    // Objects
    document.write(typeof {name: "John", age: 18} + "<br>");  // Prints: "object"

    // Arrays
    document.write(typeof [1, 2, 4] + "<br>");  // Prints: "object"

    // Functions
    document.write(typeof function(){});  // Prints: "function"
    </script>
</body>
</html>
```

```
number
number
number
number
number
string
string
string
boolean
boolean
undefined
undefined
undefined
object
object
function
```

Co-funded by the Erasmus+ Programme of the European Union

Operadores de JavaScript

- Os operadores são símbolos ou palavras-chave que informam o mecanismo de JavaScript a realizar alguma ação. Por exemplo, o símbolo de adição (+) é um operador que diz ao motor JavaScript para introduzir duas variáveis ou valores, no entanto os sinais igual (==), maior que (>) ou menor que (<) são os operadores que dizem ao JavaScript para comparar duas variáveis, valores, etc..

- Entre os diferentes operadores usados em JavaScript, os primeiros a serem descritos são os Operadores Aritméticos do JavaScript. Estes são colocados em ação de forma a realizarem operações aritméticas comuns (adições, subtrações, multiplicação e tudo o resto).

Co-funded by the
Erasmus+ Programme
of the European Union

Operadores de JavaScript

Operadores Aritméticos

Operador	Descrição	Exemplo	Resultado
+	Adição	x + y	A soma de x e y.
-	Subtração	x - y	A diferença entre x e y.
*	Multiplicação	x * y	O produto de x com y.
/	Divisão	x / y	Quociente de x e y.
%	Modulus	x % y	O resto de x dividido por y.

Co-funded by the Erasmus+ Programme of the European Union

Operadores de JavaScript

Operadores Aritméticos

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Arithmetic Operators</title>
</head>
<body>
    <script>
    var x = 10;
    var y = 4;
    document.write(x + y); // Prints: 14
    document.write("<br>");

    document.write(x - y); // Prints: 6
    document.write("<br>");

    document.write(x * y); // Prints: 40
    document.write("<br>");

    document.write(x / y); // Prints: 2.5
    document.write("<br>");

    document.write(x % y); // Prints: 2
    </script>
</body>
</html>
```

```
14
6
40
2.5
2
```

Co-funded by the
Erasmus+ Programme
of the European Union

Operadores de JavaScript

Operadores de Linhas

Operador	Descrição	Exemplo	Resultado
+	Concatenação	str1 + str2	Concatenação de str1 e str2
+=	Atribuição de Concatenação	str1 += str2	Apêndice de str2 para str1

Co-funded by the Erasmus+ Programme of the European Union

Operadores de JavaScript

Operadores de Linhas

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript String Operators</title>
</head>
<body>
    <script>
    var str1 = "Hello";
    var str2 = " World!";

    document.write(str1 + str2 + "<br>"); // Outputs: Hello World!

    str1 += str2;
    document.write(str1); // Outputs: Hello World!
    </script>
</body>
</html>
```

Hello World!
Hello World!

Co-funded by the
Erasmus+ Programme
of the European Union

Operadores de JavaScript

Os Operadores de incremento e decremento do JS são usados para incrementar/decrementar o valor de uma variável.

Operador	Nome	Efeito
++x	Pré-incremento	Incrementa x por um, depois retoma a x
x++	Pós-incremento	Retoma a x, depois incrementa x por um
--x	Pré-decremento	Decrementa x por um, depois retoma a x
x--	Pós-decremento	Retorna a x, depois decrementa x por um

Co-funded by the Erasmus+ Programme of the European Union

Operadores de JavaScript

Operadores de Incremento e Decremento de JS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Incrementing and Decrementing Operators</title>
</head>
<body>
    <script>
    var x; // Declaring Variable

    x = 10;
    document.write(++x); // Prints: 11
    document.write("<p>" + x + "</p>");   // Prints: 11

    x = 10;
    document.write(x++); // Prints: 10
    document.write("<p>" + x + "</p>");   // Prints: 11

    x = 10;
    document.write(--x); // Prints: 9
    document.write("<p>" + x + "</p>");   // Prints: 9

    x = 10;
    document.write(x--); // Prints: 10
    document.write("<p>" + x + "</p>");   // Prints: 9
    </script>
</body>
</html>
```

```
11
11
10
11
9
9
10
9
```

Co-funded by the
Erasmus+ Programme
of the European Union

Operadores de JavaScript

Operadores Lógicos

Operador	Nome	Exemplo	Resultado
&&	E	x && y	Verdadeiro se ambos x e y forem verdadeiros
||	Ou	x || y	Verdadeiro se o x ou y for verdadeiro
!	Não	!x	Verdadeiro se x não for verdadeiro

Co-funded by the Erasmus+ Programme of the European Union

Operadores de JavaScript

Operadores Lógicos

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Logical Operators</title>
</head>
<body>
    <script>
    var year = 2018;

    // Leap years are divisible by 400 or by 4 but not 100
    if((year % 400 == 0) || ((year % 100 != 0) && (year % 4 == 0))){
        document.write(year + " is a leap year.");
    } else{
        document.write(year + " is not a leap year.");
    }
    </script>
</body>
</html>
```

2018 is not a leap year.

Co-funded by the
Erasmus+ Programme
of the European Union

Operadores de JavaScript

Operadores de Comparação

Operador	Nome	Exemplo	Resultado
==	Igual	x == y	Verdadeiro se x for igual a y.
===	Idêntico	x === y	Verdadeiro se x for igual a y e se eles são do mesmo tipo.
!=	Não igual	x != y	Verdadeiro se o x não for igual ao y.
!==	Não identicos	x !== y	Verdadeiro se o x não for igual ao y ou se não forem do mesmo tipo.
<	Menor do que	x < y	Verdadeiro se x for menor que y.
>	Maior do que	x > y	Verdadeiro se o x for maior do que o y.
>=	Maior ou igual do que	x >= y	Verdadeiro se o x for maior ou igual do que y.
<=	Menor ou igual do que	x <= y	Verdadeiro se o x for menor ou igual do que o y.

Co-funded by the Erasmus+ Programme of the European Union

Operadores de JavaScript

Operadores de Comparação

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <title>JavaScript Comparison Operators</title>
6  </head>
7  <body>
8      <script>
9      var x = 25;
10     var y = 35;
11     var z = "25";
12
13     document.write(x == z);  // Prints: true
14     document.write("<br>");
15
16     document.write(x === z); // Prints: false
17     document.write("<br>");
18
19     document.write(x != y);  // Prints: true
20     document.write("<br>");
21
22     document.write(x !== z); // Prints: true
23     document.write("<br>");
24
25     document.write(x < y);   // Prints: true
26     document.write("<br>");
27
28     document.write(x > y);   // Prints: false
29     document.write("<br>");
30
31     document.write(x <= y);  // Prints: true
32     document.write("<br>");
33
34     document.write(x >= y);  // Prints: false
35     </script>
36 </body>
37 </html>
```

```
true
false
true
true
true
false
true
false
```

Co-funded by the
Erasmus+ Programme
of the European Union

Eventos de JavaScript

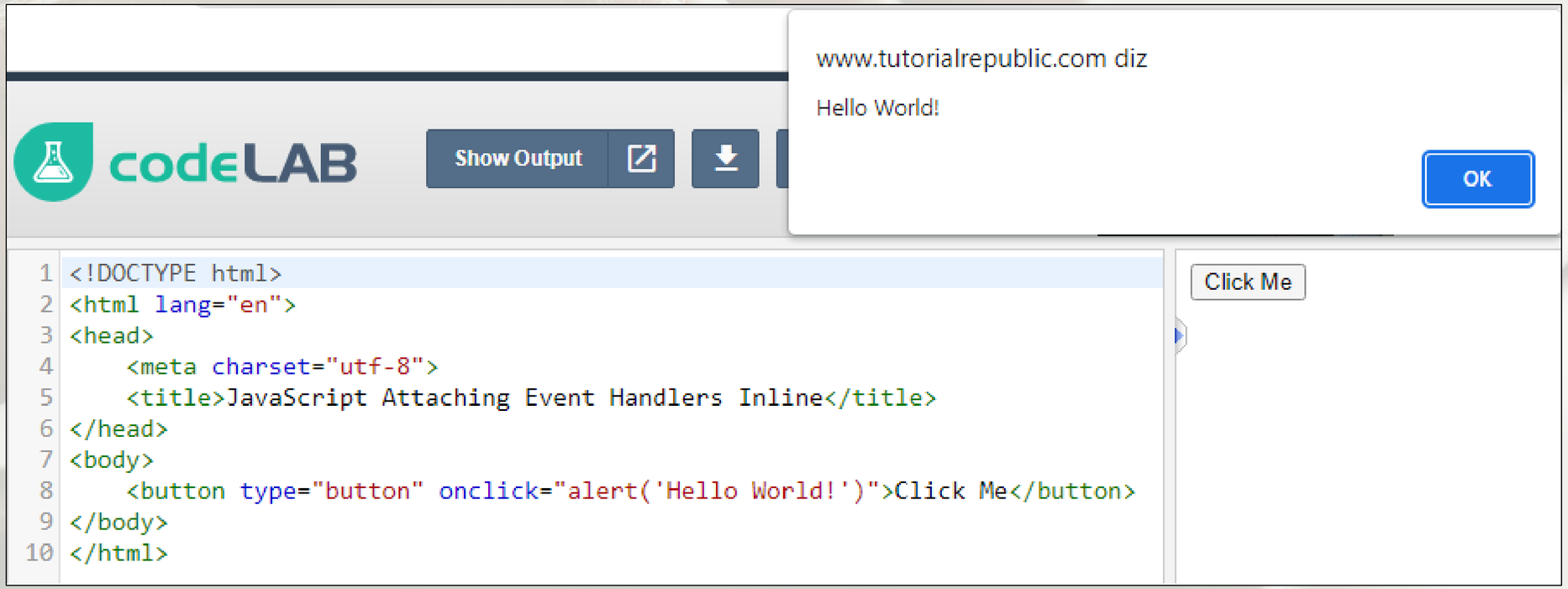
- Antes de aprofundarmos esta secção, é importante saber o que é, neste contexto, um evento. Um evento é algo que ocorre cada vez que um utilizador interage com a página web, através, por exemplo, do clique de um link ou de um botão, o texto dá entrada numa caixa de contributo ou uma área de texto, a seleção é feita na caixa de seleção, a tecla é carregada no teclado, o cursor do rato é movido, um formulário é submetido, etc. De vez em quando, o browser é capaz de desencadear o evento por si próprio, por exemplo, quando está a carregar uma página.

- Quando se dá um evento, os programadores podem usar um manipulador (ou ouvinte) de eventos JavaScript de forma e identifica-los e fazer tarefas especificas. Através da conversão, os nomes para os manipuladores de eventos começam sempre pela palavra “on”, para que um manipulador de um evento de click seja apelidado de “onclick”, da mesma forma, um manipulador para um evento de load é chamado de onload, um manipulador para o evento de blur é chamado de onblur, etc.

Co-funded by the
Erasmus+ Programme
of the European Union

Eventos de JavaScript

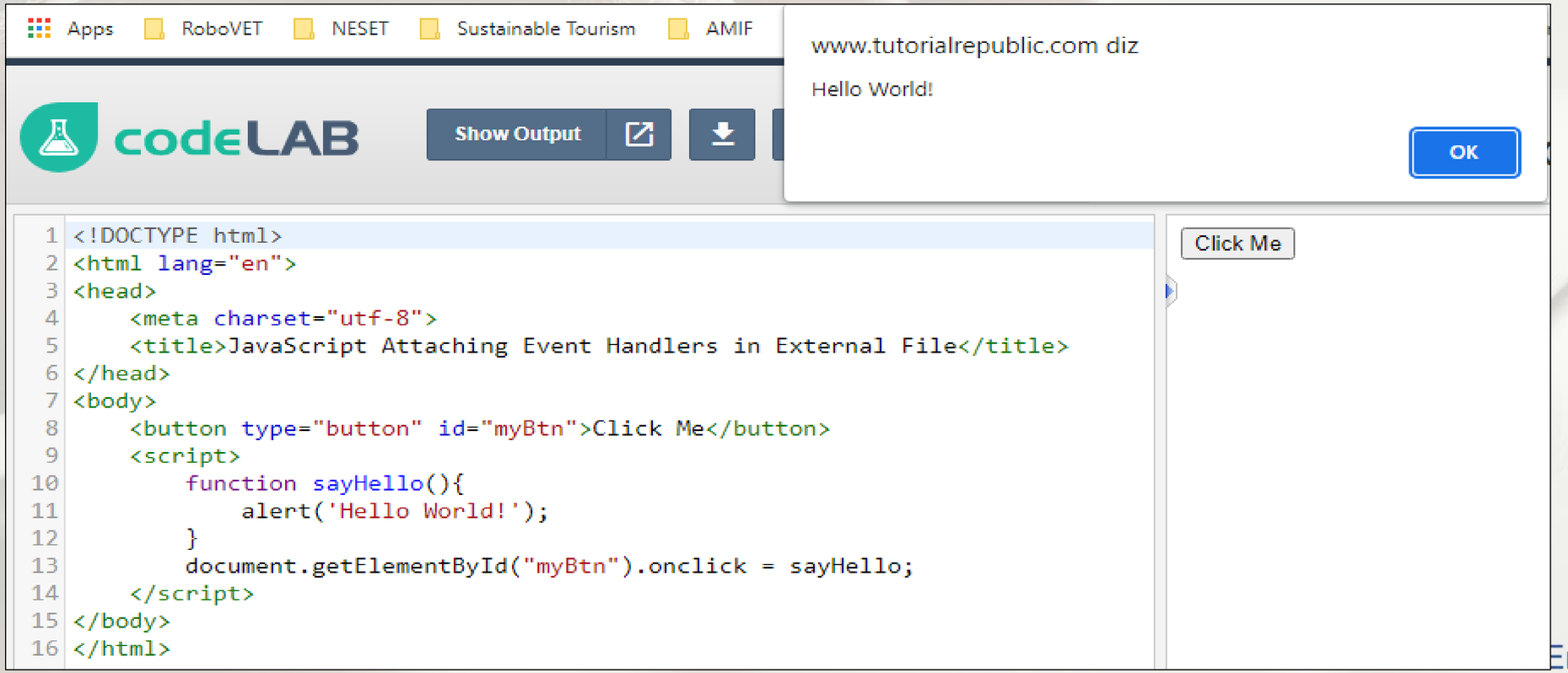
- Existem várias formas de atribuir um manipulador de eventos. A forma mais simples de adicioná-los diretamente é no início da tag de um elemento HTML, pelos atributos especiais de um manipulador de eventos. P.e.: atribuir um manipulador de clique a um elemento de botão, tal como se segue:



Co-funded by the
Erasmus+ Programme
of the European Union

Eventos de JavaScript

- No entanto, para manter o JavaScript separado do HTML, os programadores podem definir um manipulador de eventos num ficheiro de JavaScript externo ou dentro das tags <script> e </script>, tal como no seguinte exemplo:




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Attaching Event Handlers in External File</title>
</head>
<body>
    <button type="button" id="myBtn">Click Me</button>
    <script>
        function sayHello(){
            alert('Hello World!');
        }
        document.getElementById("myBtn").onclick = sayHello;
    </script>
</body>
</html>
```

Co-funded by the Erasmus+ Programme of the European Union

Eventos de JavaScript

Geralmente, os eventos podem ser categorizados em quatro grupos principais – **eventos de rato, eventos de teclado, eventos de formulário e eventos de documento/janela**.

Co-funded by the Erasmus+ Programme of the European Union

Eventos de JavaScript

- **Eventos de rato**
- Um evento de rato é ativado quando um utilizador carrega num elemento, move o cursor do rato sobre um elemento, e afins. Alguns eventos de rato importantes e os seus manipuladores de eventos são os seguintes:
  - **O Evento do Clique (onclick):** O evento do clique acontece quando um utilizador carrega num elemento de uma página web. Por norma, estes são elementos de formulário. Um evento de clique pode ser manipulado com um manipulador onclick de um evento.
  - **O Evento Contextmenu (oncontextmenu):** ocorre quando o utilizador clica, com o botão direito do rato num elemento, abrindo um menu contextualizado.
  - **O Evento Mouseover (onmouseover):** acontece quando o utilizador move o cursor do rato por cima de um elemento. Pode ser manipulado com o manipulador do evento onmouseover.
  - **O Evento Mouseout (onmouseout):** acontece quando o utilizador move o cursor do rato para fora de um elemento. Pode ser manipulado ao usar o manipulador de eventos onmouseout..

Co-funded by the Erasmus+ Programme of the European Union

Eventos de JavaScript

- **Eventos de teclado**

Um evento de teclado acontece quando o utilizador carrega ou solta uma tecla no teclado. Alguns dos eventos de teclado mais importantes e os seus manipuladores de evento são os seguintes:

- **O Evento Keydown (onkeydown):** acontece quando um utilizador carrega numa tecla de um teclado. Pode ser manipulado ao usar o manipulador do evento onkeydown.
- **O Evento Keyup (onkeyup):** ocorre quando um utilizador solta uma tecla no teclado. Manipulado pelo manipulador de evento onkeyup.
- **O Evento Keypress (onkeypress):**acontece quando um utilizador carrega numa tecla do teclado que tem um valor de caracter ligado a si. P.e.: Ctrl, Shift, Alt, Esc, Teclas de setas, etc,., não irá gerar um evento keypress, mas irá gerar um evento keydown e keyup. O manipulador do evento onkeypress manipula o evento keypress.

Co-funded by the
Erasmus+ Programme
of the European Union

Eventos de JavaScript

- **Eventos de Formulário**

  Um evento de formulário é desencadeado quando o controlo de formulário receber ou perder focos ou quando um utilizador modifica um valor de controlo do formulário (ex.: ao escrever texto numa entrada de texto), seleciona uma opção numa caixa de seleção, etc.

  - **O Evento Focus (onfocus):** acontece quando o utilizador atribui foco a um elemento de uma página web.
  - **O Evento Blur (onblur):** acontece quando o utilizador retira o foco de uma janela ou de um elemento. Este pode ser manipulado com o mainipulador de eventos onblur.
  - **O Evento Change (onchange):** acontece logo após o utilizador mudar o valor de um elemento de formulário. O manipulador do evento: onchange.
  - **O Evento Submit (onsubmit):** acontece apenas quando o utilizador submete um formulário numa página web. Manipulador de evento: onsubmit.

Co-funded by the
Erasmus+ Programme
of the European Union

Eventos de JavaScript

- **Eventos de Documentos/Janela**

Situações onde a página foi carregada ou redimensionada no browser da janela podem, também, despoletar eventos.

- **O Evento Load (onload):** acontece quando uma página web é completamente carregada num browser da web. Manipulador de evento: onload.
- **O Evento Unload (onunload):** quando o utilizador sai da webpage atual, este evento acontece. Manipulador do evento: onunload.
- **O Evento Resize (onresize):** acontece quando o utilizador de internet redimensiona, minimiza ou maximiza a janela do browser. Manipulador de evento: onresize.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Em JavaScript, as linhas representam um papel principal na estrutura geral de uma página web, pois estas são uma sequência de letras, números, caracteres especiais e valores aritméticos ou até mesmo uma combinação de todos estes. Elas podem ser criadas pelo desenrolar de uma linha literal (p.e.: linha de caracteres) quer estejam dentro de aspas únicas (‘) ou aspas duplas (‘’), como no exemplo:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Creating Strings in JavaScript</title>
</head>
<body>
    <script>
    // Creating variables
    var myString = 'Hello World!'; // Single quoted string
    var myString = "Hello World!"; // Double quoted string

    // Printing variable values
    document.write(myString + "<br>");
    document.write(myString);
    </script>
</body>
</html>
```

Hello World!
Hello World!

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

As aspas podem ser usadas dentro de uma linha, mas estas não devem corresponder às que rodeiam as linhas:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Using Quotes inside JavaScript Strings</title>
</head>
<body>
    <script>
    // Creating variables
    var str1 = "it's okay";
    var str2 = 'He said "Goodbye"';
    var str3 = "She replied 'Calm down, please'";

    // Printing variable values
    document.write(str1 + "<br>");
    document.write(str2 + "<br>");
    document.write(str3);
    </script>
</body>
</html>
```

```
it's okay
He said "Goodbye"
She replied 'Calm down, please'
```

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

No entanto, aspas únicas podem ser postas dentro de uma linha com aspas únicas ou aspas duplas dentro de linhas com aspas duplas, através da separação das aspas com uma barra inclinada para trás, tal como mostra a figura 36. A barra inclinada para trás é um termo que representa um caracter de saída e as sequencias \’ e \” são sequências de saída.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Escaping Quotes inside JavaScript Strings</title>
</head>
<body>
    <script>
    // Creating variables
    var str1 = 'it\'s okay';
    var str2 = "He said \"Goodbye\"";
    var str3 = 'She replied \'Calm down, please\'';

    // Printing variable values
    document.write(str1 + "<br>");
    document.write(str2 + "<br>");
    document.write(str3);
    </script>
</body>
</html>
```

```
it's okay
He said "Goodbye"
She replied 'Calm down, please'
```

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

As sequências de saída são valiosos para adicionar caracteres que não podem ser inseridos através do teclado. Algumas das outras sequências de saída mais utilizadas são:

- \n é substituído pelo caracter numa nova linha
- \t é substituído pela tab do caracter
- \r é substituído pelo caracter de carriage-return
- \b é substituído pelo caracter de backspace
- \\ é substituído pela barra única (\)

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Realizar Operações em Linhas

O JavaScript faz com que várias propriedades e métodos possam fazer operações em valores de linha.

- Com efeito, vários objetos podem ter propriedades e métodos. No entanto, em JavaScript, tipos de dados primitivos podem comportar-se como objetos quando o programador se refere a estes com notação de acesso à propriedade.
- O JavaScript cede esta possibilidade através da criação de um objeto embrulhado provisório para tipos de dados primitivos. Este procedimento pode ser feito automaticamente pelo intérprete JS no background.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Definir o Comprimento de uma Linha

A propriedade do comprimento define o comprimento de uma linha, que é o número de caracteres delimitados numa linha, incluindo o número de caracteres especiais, como /t ou /n. Os programadores devem ter especial atenção ao usarem parenteses depois do comprimento (p.e.: str.length()), pois a forma correta é str.length (se não for esta, irá dar erro).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Get String Length in JavaScript</title>
</head>
<body>
    <script>
    var str1 = "This is a paragraph of text.";
    document.write(str1.length + "<br>"); // Prints 28

    var str2 = "This is a \n paragraph of text.";
    document.write(str2.length); // Prints 30, because \n is only one
character
    </script>
</body>
</html>
```

28
30

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Descobrir uma Linha Dentro de Outra Linha

O método indexOf() pode ser usado para descobrir uma sublinha ou uma linha dentro de outra linha. Esta técnica devolve o índex ou a posição do primeiro incidente de uma linha dentro de uma linha específica.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Find the Position of Substring within a String</title>
</head>
<body>
    <script>
    var str = "If the facts don't fit the theory, change the facts.";
    var pos = str.indexOf("facts");
    document.write(pos); // 0utputs: 7
    </script>
</body>
</html>
```

```
7
```

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Procurar um Padrão Dentro de uma Linha

Este método de procura pode ser usado para encontrar uma peça de texto ou padrão particular dentro de uma linha. Tal como o indexOf(),search() também volta ao índex da primeira combinação, e devolve -1 se nenhum par for encontrado, mas, ao contrario do indexOf(),search () pode também usar as expressões regulares como um argumento para entregar aptidões de busca avançadas.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Find the Position of Substring within a String</title>
</head>
<body>
    <script>
    var str = "If the facts don't fit the theory, change the facts.";
    var pos = str.lastIndexOf("facts");
    document.write(pos); // 0utputs: 46
    </script>
</body>
</html>
```

```
46
```

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Procurar um Padrão Dentro de uma Linha

Este método de procura pode ser usado para encontrar uma peça de texto ou padrão particular dentro de uma linha. Tal como o indexOf(), search() também volta ao índex da primeira combinação, e devolve -1 se nenhum par for encontrado, mas, ao contrario do indexOf(), search() pode também usar as expressões regulares como um argumento para entregar aptidões de busca avançadas. Deve ser notado que o método de search() não funciona com buscas globais, pois ignora a bandeira ou modificador g no argumento regular de expressão.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Procurar um Padrão Dentro de uma Linha

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Search Text or Pattern inside a String</title>
</head>
<body>
    <script>
    var str = "Color red looks brighter than color blue.";

    // Case sensitive search
    var pos1 = str.search("color");
    document.write(pos1 + "<br>"); // 0utputs: 30

    // Case insensitive search using regexp
    var pos2 = str.search(/color/i);
    document.write(pos2); // 0utputs: 0
    </script>
</body>
</html>
```

```
30
0
```

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Extrair uma Sublinha a uma Linha

Para extrair uma parte ou uma sublinha de uma linha, pode usar-se o método slice(). Isto requer dois parâmetros: start index (onde começa a extração) e, opcionalmente, index end (índex encontrado no término da extração), como str.slice(startIndex, endIndex).

O exemplo abaixo corta uma porção de uma linha da posição 4 para a posição 15:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Slice Out a Portion of a String</title>
</head>
<body>
    <script>
    var str = "The quick brown fox jumps over the lazy dog.";
    var subStr = str.slice(4, 15);
    document.write(subStr); // Prints: quick brown
    </script>
</body>
</html>
```

quick brown

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Extrair uma Sublinha a uma Linha

Os valores negativos também podem ser específicados. Estes valores são tratados como strLength + startIndex, onde strLength é o comprimento da linha (como no caso de str.length), por exemplo, de o startIndex é -5, este é tratado como strLength – 5. Se o startIndex é maior ou igual do que o comprimento da linha, o método slice() retoma uma linha vazia. Correspondentemente, se o endIndex opcional não for especificado ou for omitido, o método slice() extrai-se para o fim da linha.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Extrair uma Sublinha a uma Linha

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Slice Strings Using Negative Indexes</title>
</head>
<body>
    <script>
    var str = "The quick brown fox jumps over the lazy dog.";
    document.write(str.slice(-28, -19) + "<br>"); // Prints: fox jumps
    document.write(str.slice(31)); // Prints: the lazy dog.
    </script>
</body>
</html>
```

fox jumps
the lazy dog.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Extrair uma Sublinha a uma Linha

O método de substring() para extrair a secção dada de uma linha baseada nos indexes de inicio e de fim, como str.substring(startIndex, endIndex). O método de substring() é bastante comparável ao do slice(), exceto algumas diferenças:

- Se algum dos argumentos for menor do que 0 ou for NaN, este é tratado como 0.
- Se algum dos argumentos for maior do que str.length, este é tratado como se fosse str.length.
- Se o startIndex for maior do que o endIndex, então o substring() vai trocar esses dois argumentos, p.e.: str.substring(5, 0) == str.substring(0, 5).

Co-funded by the Erasmus+ Programme of the European Union

Linhas em JavaScript

Extrair um Número Fixo de Caracteres de uma Linha

O JavaScript também fornece a técnica substr(), que é parecida à do slice() com uma pequena diferença: o segundo parâmetro estipula o numero de caracteres a extrair em vez do índex fina, como str.substr(startIndex, length). Se o comprimento for 0 ou um número negativo, é devolvida uma linha vazia.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Extrair um Número Fixo de Caracteres de uma Linha

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Extract Fixed Number of Characters from a
String</title>
</head>
<body>
    <script>
    var str = "The quick brown fox jumps over the lazy dog.";
    document.write(str.substr(4, 15) + "<br>"); // Prints: quick brown fox
    document.write(str.substr(-28, -19) + "<br>"); // Prints nothing
    document.write(str.substr(-28, 9) + "<br>"); // Prints: fox jumps
    document.write(str.substr(31)); // Prints: the lazy dog.
    </script>
</body>
</html>
```

```
quick brown fox

fox jumps
the lazy dog.
```

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Substituir o Conteúdo de uma Linha

A técnica de replace() é usada para substituir uma parte de uma linha por outra linha. Esta abordagem usa uma expressão regular para combinar ou uma sublinha com dois parâmetros e uma linha de substituição, p.e.: str.replace(regexp|substr, newSubstr). O método replace() devolve uma nova linha, que não interfere com a linha original que se manterá intacta.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Substituir o Conteúdo de uma Linha

Por norma, a técnica replace() substitui apenas a primeira combinação e é sensível a letras maiúsculas e minúsculas. Para substituir a sublinha dentro de uma linha numa forma sensível a maiúsculas e minúsculas, a regular *expression (regexp)* com um modificador i pode ser usada.

- Para substituir todas as incidências de uma sublinha dentro de uma linha de uma forma sensível a maiúsculas e minúsculas, podem ser usados o modificador g com o modificador i.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Transformar uma Linha para Maiúsculas ou Minúsculas

- O método toUpperCase() é usado para converter uma linha para maiúsculas.
- Da mesma forma, o método toLowerCase() é usado para converter uma linha em minúsculas.

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Concatenação de Duas ou Mais Linhas

- Duas ou mais linhas podem ser concatenadas ou combinadas através do uso de operadores atribuídos + e +=.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Join Two or More Strings</title>
</head>
<body>
    <script>
    var hello = "Hello";
    var world = "World";
    var greet = hello + " " + world;
    document.write(greet + "<br>"); // Prints: Hello World

    var wish  = "Happy";
        wish += " New Year";
    document.write(wish); // Prints: Happy New Year
    </script>
</body>
</html>
```

Hello World
Happy New Year

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Aceder a Caracteres Individuais de uma Linha

O método charAt() pode ser usado para aceder a caracteres individuais de uma linha, como str.charAt(index). O index especifico deve ser um número entre 0 e str.length - 1. Se não for atribuído qualquer index, o primeiro caracter da linha é devolvido, pois o valor base é 0.

No entanto, existem boas alternativas para este procedimento. Desde ECMAScript 5, as linhas podem ser tratadas como matrizes de leitura, e caracteres individuais podem ser vistos de uma linha usando parenteses retos ([]) em vez da técnica de charAt().

Co-funded by the
Erasmus+ Programme
of the European Union

Linhas em JavaScript

Dividir uma Linha numa Matriz

O método de split() pode ser usado para fragmentar uma linha numa matriz de linhas, usando a sintaxe str.split(separator, limit). O argumento de separador estipula qual a linha onde deve acontecer a divisão. Se o argumento de separador for omitido ou não for encontrado numa linha específica, a linha completa é alocada para o primeiro elemento da matriz.

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Existem dois tipos de números em JavaScript:

- **Números regulares** em JavaScript são armazenados num formato de 64-bit IEEE-754, também conhecido como "*double precision floating point numbers*". Estes são o tipo de números mais comuns e mais usados.

- **Números BigInt**, representam valores inteiros aleatórios de comprimento. Estas são necessários, pois um número regular não pode exceder, com segurança, $\mathbf{2^{53}}$ ou menos do que $\mathbf{-2^{53}}$.

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

- Existem mais formas de escrever um número. Por exemplo, a forma mais obvia de escrever 1 milhão seria ‘1000000000’ or ‘1_000_000_000’, usando o *underscore* como separador. Neste caso, o underscore é um “*syntactic sugar*”, que torna o número mais fácil de ler. O mecanismo do JS ignora os *underscores* entre os dígitos, sendo o mesmo 1 milhão do primeiro caso.

- No entanto, na vida real, toda a gente irá evitar escrever longas sequências de zeros. Algo como “1bn” para um milhão ou “2.5bn” para dois milhões e 500 mil parece bastante mais razoável. Os mesmos princípios aplicam-se para a maioria dos números grandes. Posto isto, é possível encurtar um número em JS ao adicionar a letra “e” ao mesmo e a especificar a quantidade de zeros.

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

```
1  let billion = 1e9;  // 1 billion, literally: 1 and 9 zeroes
2
3  alert( 7.3e9 );  // 7.3 billions (same as 7300000000 or 7_300_000_000)
```

- Portanto, o “e” multiplica o número por 1 com o número de zeros atribuídos.

```
1e3 === 1 * 1000; // e3 significa *1000
1.23e6 === 1.23 * 1000000; // e6 significa *1000000
```

- Este mesmo princípio aplica-se, também, a números pequenos. Por exemplo, o que deve ser escrito para 1 microssegundo (um milhão de um segundo)?

```
let mcs = 0.000001;
```

- Tal como o que foi mencionado antes com os números grandes, usar o "e" pode ser benéfico. Para evitar escrever os zeros explicitamente, pode ser escrito o seguinte:

```
let mcs = 1e-6; // seis zeros à esquerda do 1
```

- Existem 6 zeros em 0.000001. Então, é simples concluir que 1e-6.

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Por consequência, um número negativo antes do “e” implica a divisão por 1 com o número de zeros atribuídos, como no exemplo:

```
1  // -3 divides by 1 with 3 zeroes
2  1e-3 === 1 / 1000; // 0.001
3
4  // -6 divides by 1 with 6 zeroes
5  1.23e-6 === 1.23 / 1000000; // 0.00000123
```

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Números hex, binários e octal

Números hexadecimais são comuns no JavaScript para a atribuição das cores, codificação de caracteres e muitos outros propósitos. Portanto, obviamente, existe uma forma mais rápida de escrevê-los: 0x e depois o número, tal como aqui:

```
1  alert( 0xff ); // 255
2  alert( 0xFF ); // 255 (the same, case doesn't matter)
```

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Números hex, binários e octal

- Sistemas numerais binários e octais não são utilizados com muita frequência, mas são também compatíveis através do uso dos prefixos de 0b e 0o (zero, o).
- Existem apenas 3 sistemas numerais com este suporte. Para mais sistemas numerais deve ser usado a função parseInt.

Co-funded by the Erasmus+ Programme of the European Union

Números em JavaScript

Método toString(base)

O método num.toString(base) devolve uma linha de representação de um sistema numérico com as bases providenciadas, como no exemplo:

```
1  let num = 255;
2
3  alert( num.toString(16) );  // ff
4  alert( num.toString(2) );   // 11111111
```

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Método toString(base)

- Existe um caso específico que deve ser considerado. Quando dois pontos são localizados em 123456..toString(36), isto não é uma gralha. Quando o programador quer aplicar um método diretamente a um número, é preciso colocar os dois pontos (‘..’) de seguida.
- Se apenas for colocado um único ponto(‘) 123456.toString(36’), isto seria um erro, pois a sintaxe de JavaScript determina a parte decimal após o primeiro ponto. E, se o programador colocar mais um ponto, então o JavaScript sabe que a parte decimal está vazia e procede para o método.

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Arredondamento

Uma das operações mais aplicadas aquando da operação de números é o **arredondamento**.

Aqui estão algumas funções *built-in* para **arredondamento**:

- **Math.floor**

Arredonda para baixo: 3.1 fica 3, e -1.1 fica -2.

- **Math.ceil**

Arredonda para cima: 3.1 fica 4, e -1.1 fica -1.

- **Math.round**

Arredonda para o número inteiro mais próximo: 3.1 fica 3, 3.6 fica 4, no caso do meio: 3.5 fica 4 também.

- **Math.trunc (não suportado pelo Internet Explorer)**

Elimina tudo após o ponto decimal sem arredondar: 3.1 fica 3, -1.1 arredonda para -1.

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Arredondamento

As funções presentes no último slide cobrem todas as formas possíveis para lidar com a parte decimal de um número. Mas então como é que se arredonda um número ao dígito n-th após o decimal?

Por exemplo, para arredondar 1.2345 em dois dígitos (1.23).

Estas são as duas maneiras de o fazer:

1. **Multiply-and-divide.**

P.ex., para arredondar o número ao 2º digíto depois do décimal, podemos multiplicar o número por 100 (ou uma potência maior de 10), aplicar a função de arredondamento e depois dividi-lo novamente.

2. O método **toFixed(n)** arredonda o número para n dígitos depois do ponto e retoma a representação do resultado numa linha.

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Cálculos Imprecisos

No interior, um número é representado num formato 64-bit **IEEE-754**, o que quer dizer que existem precisamente 64 bits para armazenar um número; 52 para armazenar os dígitos; 11 para armazenar a posição do ponto decimal (zero para os números inteiros), e 1 bit para o sinal.

- Se o número for muito longo, irá ultrapassar o armazenamento de 64-bits, dando, possivelmente, infinito.

- Algo que acaba por acontecer com alguma frequência é a perda de precisão.

Co-funded by the
Erasmus+ Programme
of the European Union

Números em JavaScript

Cálculos Imprecisos

- O método mais fiável para lidar com esta situação é o **toFixed(n)**:

```
1  let sum = 0.1 + 0.2;
2  alert( sum.toFixed(2) ); // 0.30
```

Co-funded by the
Erasmus+ Programme
of the European Union

Testes: isFinite e isNaN

Anteriormente, estes dois valores numéricos especiais foram descritos.

- **Infinity** (e **-Infinity**) são um valor número especial que é maior (menor) que alguma coisa.
- **NaN** representa um erro.

Co-funded by the Erasmus+ Programme of the European Union

Números em JavaScript

Testes: isFinite e isNaN

Ambos pertecem ao tipo número, mas não são números “normais”, então existem funções especiais para estes:

- **isNaN(value)** converte o argumento para um número e depois testa se é NaN:

```
1  alert( isNaN(NaN) ); // true
2  alert( isNaN("str") ); // true
```

- **isFinite(value)** converte o seu argumento num número e fará um retorno como sendo **verdadeiro** se se tratar de um número regular, e não **NaN/Infinity/-Infinity**.

```
1  alert( isFinite("15") ); // true
2  alert( isFinite("str") ); // false, because a special value: NaN
3  alert( isFinite(Infinity) ); // false, because a special value: Infinity
```

Números em JavaScript

parseInt e parseFloat

- A conversão numérica usando um mais (**+**) ou Number() é rigorosa. Se o valor não for exatamente um número, falha.
- **parseInt** e **parseFloat** destinam-se a extrair valores numéricos destas situações inesperadas (por ex., quando os símbolos aparecem depois do número – 18€).
- Eles "*lêem*" um número de uma linha até não poderem. Em caso de erro, o número recolhido é devolvido. A função **parseInt** retoma um número inteiro, onde o **parseFloat** devolve um número em ponto-flutuante.

Co-funded by the Erasmus+ Programme of the European Union

Números em JavaScript

Outras funções matemáticas

O JavaScript é detentor de um *built-in* objeto Math, que inclui uma pequena biblioteca de funções e constantes matemáticas:

- **Math.random()**

  Devolve um número aleatório do 0 ao 1 (não incluindo o 1).

- **Math.max(a, b, c...) / Math.min(a, b, c...)**

  Devolve o maior/menor valor do número arbitrário de argumentos.

- **Math.pow(n, power)**

  Devolve *n* elevado à potência dada.

Co-funded by the Erasmus+ Programme of the European Union

Expressões JavaScript If…Else

Tal como outras linguagens de programação, o JavaScript também permite escrever código que execute ações diferentes baseadas nas condições de testes lógicos ou comparativos na altura em que estes foram corridos. Assim, as condições de teste podem ser criadas como expressões que avaliam como verdadeiro ou falso e, baseado nestes resultados, certas ações podem ser efetuadas.

Existem várias expressões de condições em JavaScript que podem ser usadas para tomar decisões:

- A expressão **if** ;
- A expressão **if...else**;
- A expressão **if...else if....else**;
- A expressão **switch...case**.

Co-funded by the
Erasmus+ Programme
of the European Union

Frases JavaScript If…Else

A expressão “if”

A expressão “if” é usada para executar um bloco de código apenas se a condição específica de avaliação for verdadeira. Esta é a frase condicional mais simples de JS e pode ser escrita como:

```
if(condition) {
    // Code to be executed
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript If Statement</title>
</head>
<body>
    <script>
    var now = new Date();
    var dayOfWeek = now.getDay(); // Sunday - Saturday : 0 - 6

    if(dayOfWeek == 5) {
        document.write("Have a nice weekend!");
    }
    </script>
    <p><strong>Note:</strong> This example will print "Have a nice weekend!" if the current day is Friday.</p>
</body>
</html>
```

Note: This example will print "Have a nice weekend!" if the current day is Friday.

Frases JavaScript If…Else

A expressão “if…else”

O processo de tomar decisões em JS pode ser melhorado ao adicionar uma escolha alternativa através de **expressões else** à **frase**. A expressão “if…else” permite executar um bloco de código se as condições especificas avaliadas como verdadeiras e outro bloco de código se avaliadas como *falsas*.

```
if(condition) {
    // Code to be executed if condition is true
} else {
    // Code to be executed if condition is false
}
```

Co-funded by the
Erasmus+ Programme
of the European Union

Frases JavaScript If…Else

The Ternary Operator

O operador ternário dá-nos uma **forma abreviada de escrever frases "if…else"**. É caracterizado pelo símbolo de ponto de interrogação (?) e requer três operações: uma condição para analisar, um resultado para uma "verdade" e um resultado para "falso". A sua síntaxe básica segue no seguinte exemplo:

```
var result = (condition) ? value1 : value2
```

Se a condição for avaliada como verdadeira, o value1 será devolvido, se não será devolvido o value2.

Co-funded by the
Erasmus+ Programme
of the European Union

Expressões “Switch…Case”

- A expressão “**switch..case”** é um cenário alternativo para as frases “**if...else if...else”**, que são quase o mesmo. As expressões “**switch...case”** analisam a variável ou a expressão perante uma série de valores até que encontre uma combinação, e depois executa um bloco de código correspondente para combinar. A sua sintaxe apresenta-se da seguinte forma:

```
switch(x){
    case value1:
        // Code to be executed if x === value1
        break;
    case value2:
        // Code to be executed if x === value2
        break;
    ...
    default:
        // Code to be executed if x is different from all values
}
```

Co-funded by the
Erasmus+ Programme
of the European Union

Expressões “Switch…Case”

- Este exemplo mostra o nome do dia da semana em que o utilizador está.

- A expressão “**switch...case”** diverge da expressão “**if...else”** de uma forma expressiva. A expressão “**switch”** executa linha por linha e, quando o JavaScript descobre uma cláusula de caso que avalie se é verdadeiro, efetua, não apenas o código correspondente a essa cláusula de caso, como também executa todas as cláusulas de causa automaticamente até ao fim do bloco “**switch**”.

- De forma a prever isto, deve ser incluído uma expressão de quebra depois de todos os casos. A expressão de quebra informa o intérprete JS que tem de sair do bloco da expressão de **switch...case** uma vez que execute o código associado ao primeiro caso verdadeiro.

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

As matrizes são variáveis complexas que permitem armazenar mais do que um valor ou um grupo dentro de um único nome de variável. As Matrizes do JavaScript podem ser armazenar qualquer valor válido, incluindo linhas, números, objetos, funções, e, ao mesmo tempo, outras matrizes, por consequência permitindo a criação de dados estruturados mais complexos como uma matriz de objetos ou uma matriz de matrizes. No seguinte exemplo, o nome das cores pode ser armazenado num código JavaScript:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Storing Single Values</title>
</head>
<body>
    <script>
    // Creating variables
    var color1 = "Red";
    var color2 = "Green";
    var color3 = "Blue";

    // Printing variable values
    document.write(color1 + "<br>");
    document.write(color2 + "<br>");
    document.write(color3);
    </script>
</body>
</html>
```

```
Red
Green
Blue
```

Co-funded by the Erasmus+ Programme of the European Union

Matrizes do JavaScript

Criar uma Matriz

A forma mais simples de criar uma matriz em JavaScript é de envolver uma lista de valores em parenteses retos ([]) separados por virgulas, tal como está na seguinte sintaxe:

```
var myArray = [element0, element1, ..., elementN];
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Creating Arrays in JavaScript</title>
</head>
<body>
    <script>
    // Creating variables
    var colors = ["Red", "Green", "Blue"];
    var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];
    var cities = ["London", "Paris", "New York"];
    var person = ["John", "Wick", 32];

    // Printing variable values
    document.write(colors + "<br>");
    document.write(fruits + "<br>");
    document.write(cities + "<br>");
    document.write(person);
    </script>
</body>
</html>
```

```
Red,Green,Blue
Apple,Banana,Mango,Orange,Papaya
London,Paris,New York
John,Wick,32
```

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Aceder aos elementos de uma Matriz

Os elementos de uma matriz podem ser acedidos pelo índex, utilizando a notação dos parenteses retos. Um índex é um elemento que representa a posição de um elemento numa matriz.

- As matrizes são baseadas no 0. Isto significa que o primeiro item de uma matriz é armazenada no índex 0 e não no último, pela segunda vez que os itens são armazenados no índex 2, e etc. Por entanto, uma matriz de cinco elementos terá indexes do 0 ao 4.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Access Individual Elements of an Array</title>
</head>
<body>
    <script>
    var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

    document.write(fruits[0] + "<br>"); // Prints: Apple
    document.write(fruits[1] + "<br>"); // Prints: Banana
    document.write(fruits[2] + "<br>"); // Prints: Mango
    document.write(fruits[fruits.length - 1]); // Prints: Papaya
    </script>
</body>
</html>
```

Apple
Banana
Mango
Papaya

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Looping através dos elementos de uma Matriz

O *loop* pode ser usado para ganhar acesso a cada elemento de uma matriz numa ordem sequencial, como no exemplo que se segue:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Loop Through an Array Using For Loop</title>
</head>
<body>
    <script>
    var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

    // Iterates over array elements
    for(var i = 0; i < fruits.length; i++){
        document.write(fruits[i] + "<br>"); // Print array element
    }
    </script>
</body>
</html>
```

Apple
Banana
Mango
Orange
Papaya

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Adicionar Novos Elementos a uma Matriz

To Para adicionar um novo elemento no final de uma matriz, use, simplesmente o método push(), tal como demonstrado de seguida:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Add a New Element at the End of an Array</title>
</head>
<body>
    <script>
    var colors = ["Red", "Green", "Blue"];
    colors.push("Yellow");

    document.write(colors + "<br>"); // Prints: Red,Green,Blue,Yellow
    document.write(colors.length); // Prints: 4
    </script>
</body>
</html>
```

```
Red,Green,Blue,Yellow
4
```

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Remover Elementos de uma Matriz

Para eliminar o último elemento de uma matriz podemos usar o método pop(). Este método devolve o valor que foi retirado.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Remove the Last Element from an Array</title>
</head>
<body>
    <script>
    var colors = ["Red", "Green", "Blue"];
    var last = colors.pop();

    document.write(last + "<br>"); // Prints: Blue
    document.write(colors.length); // Prints: 2
    </script>
</body>
</html>
```

```
Blue
2
```

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Adicionar ou Remover Elementos em Qualquer Posição

O método splice() é um método de matriz bastante flexível que permite adicionar ou remover elementos de qualquer sintaxe, usando `arr.splice(startIndex, deleteCount, elem1, ..., elemN)`

Este método requer três parâmetros: o primeiro é o índex onde será para iniciar a divisão da matriz, é obrigatório; o segundo é o número de elementos a remover (0 deve ser usado caso o programador não queira retirar nenhum elemento), é opcional; e o terceiro parâmetro é um conjunto de elementos de substituição, também estes opcionais.

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Adicionar ou Remover Elementos em Qualquer Posição

O método splice() devolve uma matriz aos elementos apagados, ou uma matriz vazia se não tiverem sido eliminados nenhuns elementos, tal como pode ser visto no slide anterior. Se o segundo argumento está em falta, todos os elementos desde o início até ao fim da matriz são removidos. Ao contrário dos métodos slice() e concat() o método splice() modifica a forma como a matriz é chamada.

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Criar uma Linha Através de uma Matriz

- Podem existir situações onde o programador pretende, simplesmente, criar uma linha ao juntar os elementos de uma matriz. Para isto, este pode usar o método join(). Este método retira um parâmetro opcional que é uma linha separadora que é adicionada entre cada elemento. Se omitirmos o separador, o JavaScript usará, por definição, uma vírgula (,).

- Uma matriz pode também ser convertida em linhas separadas por vírgulas ao usar toString(). Este método não permite o parâmetro de separação como join().

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Juntar Duas ou Mais Matrizes

- O método concat() pode ser usado para combinar duas ou mais matrizes. Este método não altera as matrizes prevalecentes, em vez disso, devolve uma nova matriz.
- O método concat() pode levar qualquer número de argumentos de matrizes de forma a que uma matriz possa ser criada a partir de qualquer outro número de matrizes, tal como demonstrado no seguinte exemplo:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Merge Multiple Arrays</title>
</head>
<body>
    <script>
    var pets = ["Cat", "Dog", "Parrot"];
    var wilds = ["Tiger", "Wolf", "Zebra"];
    var bugs = ["Ant", "Bee"];

    // Creating new array by combining pets, wilds and bugs arrays
    var animals = pets.concat(wilds, bugs);
    document.write(animals); // Prints: Cat,Dog,Parrot,Tiger,Wolf,Zebra,Ant,Bee
    </script>
</body>
</html>
```

```
Cat,Dog,Parrot,Tiger,Wolf,Zebra,Ant,Bee
```

Co-funded by the Erasmus+ Programme of the European Union

Matrizes do JavaScript

Procurar Através de uma Matriz

- Os métodos de indexOf() e lastIndexOf() podem ser usados para procurar, numa matriz, um valor específico. Se o valor for encontrado, ambos os métodos devolvem um índex que representa o elemento da matriz. Se o valor não for encontrado, -1 será devolvido.

- O método indexOf() devolve o primeiro encontrado, enquanto que o lastIndexOf() devolve o último. Ambos os métodos reconhecem, também, um parâmetro opcional inteiro do índex, que especifica o índex dentro da matriz na qual começou a busca.

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Procurar Através de uma Matriz

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Search an Array for a Specific Value</title>
</head>
<body>
    <script>
    var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

    document.write(fruits.indexOf("Apple") + "<br>"); // Prints: 0
    document.write(fruits.indexOf("Banana") + "<br>"); // Prints: 1
    document.write(fruits.indexOf("Pineapple")); // Prints: -1
    </script>
</body>
</html>
```

```
0
1
-1
```

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Procurar Através de uma Matriz

- O método includes() pode também ser usado para descobrir se uma matriz envolve um certo elemento ou não. Este método usa os mesmos parâmetros que os métodos indexOf() e lastIndexOf(), mas retorna resultados verdadeiros ou falsos em vez de um número índex.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Find Whether an Array Includes a Certain Value</title>
</head>
<body>
    <script>
    var arr = [1, 0, 3, 1, false, 5, 1, 4, 7];

    document.write(arr.includes(1) + "<br>"); // Prints: true
    document.write(arr.includes(6) + "<br>"); // Prints: false
    document.write(arr.includes(1, 2) + "<br>"); // Prints: true
    document.write(arr.includes(3, 4)); // Prints: false
    </script>
</body>
</html>
```

```
true
false
true
false
```

Co-funded by the
Erasmus+ Programme
of the European Union

Matrizes do JavaScript

Searching Through an Array

- To search an array based on certain condition the JavaScript find() method can be used, which has been recently introduced in ES6. This method returns the value of the first element in the array that fulfils the provided testing function. If not, it returns undefined.
- The find() method simply searches the first element that meets the provided testing function. Still, if the programmer intends to find out all the matched elements, the filter() method can be used. It creates a new array with all the elements that successfully passes the given test, as can be checked in this example.

Co-funded by the
Erasmus+ Programme
of the European Union

Ordenação de Matrizes no JavaScript

- A ordenação é uma tarefa popular quando se trabalha com matrizes.
- Pode ser usada, por exemplo, para mostrar os nomes de cidades ou países por ordem alfabética.
- O objeto de matrizes do JavaScript tem um método *built-in* – sort() – para ordenar elementos de matrizes por ordem alfabética.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Sort an Array Alphabetically</title>
</head>
<body>
    <script>
    var fruits = ["Banana", "Orange", "Apple", "Papaya", "Mango"];
    var sorted = fruits.sort();

    document.write(fruits + "<br>"); // Outputs: Apple,Banana,Mango,Orange,Papaya
    document.write(sorted); // Outputs: Apple,Banana,Mango,Orange,Papaya
    </script>
</body>
</html>
```

```
Apple,Banana,Mango,Orange,Papaya
Apple,Banana,Mango,Orange,Papaya
```

Co-funded by the Erasmus+ Programme of the European Union

Ordenação de Matrizes no JavaScript

- Para **inverter a ordem dos elementos** de uma matriz, deve ser usado o método reverse(). Inverte uma matriz de uma forma a que o primeiro elemento da matriz passa para último, e vice-versa. Os métodos sort() e reverse() alteram a matriz inicial e devolvem uma referência para a mesma matriz, tal como pode ser visto aqui.

- Para **ordenar matrizes numéricas**, não é aconselhável usar o método sort(), pois pode produzir resultados inesperados. Consequentemente, os programadores devem passar uma função comparativa, como quando uma função de comparação é especifica, os elementos de uma matriz são ordenados de acordo com o valor de retomo da função de comparação.

Co-funded by the
Erasmus+ Programme
of the European Union

Ordenação de Matrizes no JavaScript

- De forma a **descobrir o máximo e o mínimo valor numa matriz**, deve ser usado o método apply(), em combinação com o método Math.max() e Math.min(), tal como explicado aqui. O método apply() oferece uma forma acessível de transformar matrizes de valores como argumentos a funções que aceitam múltiplos argumentos de uma maneira semelhante a matrizes, mas não são matrizes. Depois, o resultado proveniente da expressão Math.max.apply(null, numbers) no exemplo acima é equivalente ao Math.max (3, -7, 10, 8, 15, 2).

- Por fim, para **ordenar uma matriz de objetos**, podemos usar o método sort(). Neste exemplo, é mostrado como ordenar uma matriz de objetos pelos valores de propriedade.

Co-funded by the
Erasmus+ Programme
of the European Union

Laços (*loops*) em JavaScript

Os laços são aplicados para repetirem o mesmo bloco de código um número de vezes, caso uma certa condição seja encontrada. A ideia principal por detrás do laço é mecanizar tarefas repetitivas dentro de um programa de forma a poupar tempo e esforço. O JavaScript suporta cinco diferentes tipos de laços:

while — laços através de um bloco de código se a condição específica avaliada for *verdadeira*.

do…while — laços através de um bloco de código uma vez; depois a condição é avaliada. Se a condição for *verdadeira*, a expressão é repetida se a condição específica for *verdadeira*.

for — laços através de um bloco de código até que o contador atinja um número específico.

for…in — laços através das propriedades de um objeto.

for…of — laços sobre objetos iteráveis como matrizes, linhas, etc.

Co-funded by the
Erasmus+ Programme
of the European Union

Laços (*loops*) em JavaScript

O laço while

Este é a expressão de laço mais fácil que é dada pelo JS. Faz laço sobre um bloco de código se uma condição específica de avaliação se verificar verdadeira. Uma vez que esta condição não se verifique, o laço é interrompido. A sintaxe geral do laço *while* é:

```
while(condition) {
    // Code to be executed
}
```

- Neste exemplo, um laço continua a correr enquanto as variáveis $i \leq 5$. Tal como se verifica, irá aumentar por 1 cada vez que o laço correr. Os programadores devem ter a certeza de que a condição especificada no laço possa, a certa altura, tornar-se negativa. Se não, o laço nunca vai parar de correr (laço infinito).

Co-funded by the Erasmus+ Programme of the European Union

Laços (*loops*) em JavaScript

O laço *do…while*

- O laço *do-while* é uma variante do laço *while*, que acede à condição no fim de cada laço gerado. Com o laço *do…while*, o bloco de código é executado uma vez, e depois a condição é avaliada. Se a condição for verdadeira, a ação é repetida se a condição específica de avaliação for verdadeira. A sintaxe comum é:

```
do {
    // Code to be executed
}
while(condition);
```

Co-funded by the
Erasmus+ Programme
of the European Union

Laços (loops) em JavaScript

O laço *do…while*

- O código JavaScript neste exemplo define um laço que começa com i=1. Irá depois produzir o resultado e aumentar o valor da variante de i por 1. Depois, esta condição será avaliada e o laço continuará a correr se i ≤ 5.

Co-funded by the
Erasmus+ Programme
of the European Union

Laços (loops) em JavaScript

O laço *for*

Repete um bloco de código se uma certa condição se verificar. É normalmente usada para implementar um bloco de código para um certo número de vezes. A sintaxe é:

```
for(initialization; condition; increment) {
    // Code to be executed
}
```

Co-funded by the
Erasmus+ Programme
of the European Union

Laços (loops) em JavaScript

O laço *for*

- O contador de laços p.e.: a variável in para o laço for-in é uma linha e não um número. Engloba o nome da propriedade atual ou o índex dos elementos da matriz em uso.

- Este exemplo demonstra como é que o laço percorre todas as propriedades de um objeto JS.

Co-funded by the
Erasmus+ Programme
of the European Union

The for...of Loop

- O ES6 introduz um novo laço *for-of* que permite a iteração sobre matrizes ou outros objetos iteráveis (p.e.: linhas) de uma forma simplificada. Para além disso, o código dentro do laço é executado para cada elemento de um objeto iterável. Este exemplo mostra como é que um laço percorre linhas e matrizes.

- Este exemplo mostra como é que um laço percorre linhas e matrizes.

Co-funded by the
Erasmus+ Programme
of the European Union

Funções do JavaScript

Uma função é um conjunto de expressãos que realizam tarefas específicas e podem ser guardadas e mantidas de forma independente. As funções permitem criar conjuntos de código reutilizável que são mais fáceis de gerir e mais simples de limpar.

Co-funded by the
Erasmus+ Programme
of the European Union

Funções do JavaScript

Estas são algumas das vantagens que se podem obter com o uso de funções:

- **Funções que diminuem a repetição de código dentro de um programa —** As funções permitem extrair blocos de código que são usados frequentemente para componentes únicos. Desta forma, é possível realizar a mesma tarefa ao invocar esta função o que for necessário para este script sem ter que copiar e colar o mesmo bloco de código vezes sem conta.
- **Funções fazem como que o código seja mais fácil de manter —** Tendo em conta que as funções, uma vez criadas, podem ser aplicadas várias vezes, quaisquer alterações feitas dentro da função são automaticamente aplicadas em todos os locais, sem que os respetivos documentos fiquem afetados.
- **Funções facilitam a eliminação dos erros —** Quando o programa é dividido em funções, o programador sabe exatamente que função é que está a gerar erro e como encontrá-la. Por consequência, corrigir erros torna-se mais simples.

Co-funded by the
Erasmus+ Programme
of the European Union

Funções do JavaScript

Definir e Nomear Funções

A declaração de uma função começa com a função no teclado, seguindo-se pela determinação do nome da função a ser criada, e depois seguem-se os parenteses e, por fim, o código da função entre chavetas {}. Para declarar uma função devemos aplicar a seguinte sintaxe:

```
function functionName() {
    // Code to be executed
}
```

Co-funded by the
Erasmus+ Programme
of the European Union

Funções do JavaScript

Definir e Nomear Funções

Neste simples exemplo exibe-se a mensagem “*Hello*”.

- Uma vez que a função seja definida, pode ser chamada por qualquer local no documento, ao escrever o nome seguido por um par de parenteses, como, por exemplo, sayHello(), usado no exemplo anterior.

Co-funded by the
Erasmus+ Programme
of the European Union

Funções do JavaScript

Adicionar Parametros às Funções

- A função displaySum() neste exemplo recebe dois números como argumentos, adiciona-os como um e depois mostra o resultado no browser. Simples. Não existe um limite na definição de parâmetros. .

- Todavia, para cada parâmetro especificado, é necessário que um argumento passe para a função quando este for chamado. Se isto não acontecer, o valor torna-se indefinido.

Co-funded by the Erasmus+ Programme of the European Union

Valores Pré-Definidos para Parâmetros de Funções

Com o ES6, é possível especificar valores pré-definidos para os parâmetros das funções. Isto implica que, se nenhum argumento for dado à função quando esta for chamada, estes valores pré-definidos serão usados. Este exemplo é bastante claro sobre o quão importante esta funcionalidade é, pois, de forma a conseguir o mesmo resultado, teríamos, anteriormente, que usar este.

Co-funded by the Erasmus+ Programme of the European Union

Funções do JavaScript

Devolver Valores de uma Função

- Uma função pode devolver um valor de novo para o script que chamou a função, como um resultado, ao utilizar a expressão de retorno. Este valor pode ser de qualquer tipo (p.e.: matrizes e objetos). A expressão de retomo é, por norma, colocada na última linha da função antes de fechar com chavetas e termina com um ponto e vírgula (;), tal como mostrado aqui.

- Uma função pode não devolver múltiplos valores. Ainda assim, resultados semelhantes podem ser obtidos ao devolver uma matriz de valores, tal como mostrado aqui.

Co-funded by the
Erasmus+ Programme
of the European Union

Funções do JavaScript

Trabalhar com Expressões de Funções

- A sintaxe usada, anteriormente, para criar funções era chamada de **declaração de funções**. Existe outra sintaxe para construir uma função - expressão de funções. Uma vez que esteja armazenada num variável, esta variável pode ser usada como função.

- A sintaxe de uma declaração de função e de uma expressão de função parecem bastante semelhantes, mas estas variam na forma em que são avaliadas. Tal como observado no exemplo anterior, a expressão de funções permitiu uma exceção quando foi chamada antes de ser definida, mas a declaração da função foi executada com eficiência.

- O JS analisa a declaração da função antes de o programa a executar. Posto isto, não faz diferença se o programa chama a função antes de esta estar definida, pois o JavaScript elevou a função para o topo do scope atual no plano de fundo. A expressão da função não foi analisada até ser atribuída a uma variável. Por consequência, esta ainda está indefinida no momento em que é chamada.

Co-funded by the Erasmus+ Programme of the European Union

Funções do JavaScript

Entender o Escopo das Variáveis

- As variáveis podem ser declaradas em qualquer local do JS, no entanto, a localização da declaração irá estabelecer a extensão da disponibilidade de uma variável dentro do programa do JavaScript – este processo pode também ser chamado de **escopo das variáveis**.

- Por defeito, as variáveis declaradas dentro de uma função têm um escopo local, o que significa que não podem ser vistas ou controladas pelo exterior dessa função, tal como mostrado aqui.

- Mesmo que quaisquer variáveis declaradas fora de uma função, num programa, têm um escopo global, independentemente de onde estiver localizada, no script, a função em caso, tal como pode ser verificado aqui.

Co-funded by the
Erasmus+ Programme
of the European Union

Objetos JavaScript

- O JavaScript é uma linguagem baseada num objeto e, ali, praticamente tudo é um objeto ou age como um. Portanto, para trabalhar corretamente com o JS, os programadores necessitam de entender como funcionam os objetos, assim como entender como criar objetos e como os usar.

- Um objeto JavaScript é simplesmente uma coleção de valores com nomes. Estes são tipicamente referidos como propriedades de um objeto. Sendo uma matriz uma coleção de valores, onde cada valor tem um índex (uma chave numérica) que começa em zero e aumenta um para cada valor. Um objeto é como uma matriz, mas a diferença é que o programador define as chaves (nome, idade, género, etc.).

Co-funded by the
Erasmus+ Programme
of the European Union

Objetos JavaScript

Criar um Objeto

- Os programadores podem criar objetos com chavetas, incluindo uma lista voluntária de propriedades. Uma propriedade pode ser um par “key:value”, onde a chave (ou nome da propriedade) é sempre uma linha, e o valor (ou valor da propriedade) pode ser qualquer tipo de dados (linhas, números, booleans, matrizes, funções, etc.).

- Para além disso, as propriedades com funções a representar os valores são, frequentemente, chamados de métodos para os distinguir de outras funcionalidades. Um objeto JS pode ser isto. Este exemplo cria um objeto chamado de pessoa, que tem 3 propriedades (nome, idade e género) e um método displayName().

Co-funded by the
Erasmus+ Programme
of the European Union

Objetos JavaScript

Creating an object

- Este método mostra o valor de this.name, que vai ao acordo do person.name. Esta é a forma ideal e mais fácil de criar um objeto novo em JS, que é conhecido como a **sintaxe literal de um objeto**.

- A propriedade de nome não precisa, geralmente, de ser citada, com excessão se forem palavras reservadas ou se contiverem espaços ou caracteres especiais (qualquer outra coisa que não letras, números e os caracteres _ e $), ou se estas começam com um número, tal como mostrado aqui..

Co-funded by the
Erasmus+ Programme
of the European Union

Objetos JavaScript

Aceder às Propriedades dos Objetos

- De forma a aceder ou ter o valor de uma propriedade, podemos aplicar o ponto (.), assim como a notação em parenteses retos ([]), tal como mostra este exemplo. A notação do ponto é fácil de ler e de escrever, mas nem sempre pode ser usada. Se o nome da propriedade não for válido (por exemplo, se tiver caracteres especiais ou espaços), a notação do ponto não pode ser usada, tendo que usar a notação de parenteses nesses casos.

- Esta mostra muita mais flexibilidade do que a notação do ponto e, para além disso, permite identificar os nomes das propriedades como variáveis de forma a substituir as literais das linhas.

Co-funded by the
Erasmus+ Programme
of the European Union

Objetos JavaScript

Laços Através das Propriedades dos Objetos

Os programadores podem iterar através de pares de valores-chave de um objeto usando o laço for...in. É especialmente valorizado por iterar sobre as propriedades dos objetos, tal como pode ser visto aqui.

Definir as Propriedades dos Objetos

Da mesma forma, podem-se definir propriedades novas ou atualizar as existentes ao usar a notação de ponto (.) ou de parenteses retos ([]), tal como demonstrado aqui.

Co-funded by the
Erasmus+ Programme
of the European Union

Objetos JavaScript

Eliminar as Propriedades dos Objetos

- O operador de eliminação pode ser aplicado para apagar, por completo, as propriedades de um objeto. Para nos livrarmos, por completo, de uma propriedade, temos obrigatoriamente de as remover.
- Ajustar a propriedade para um estado de indefinido ou nulo apenas vai alterar o seu valor, não removendo as propriedades do objeto.

- Portanto, não tem qualquer efeito em variáveis ou funções declaradas. Tendo isto em conta, os programadores devem evitar o operador de eliminação para apagar elementos de matriz, pois não altera o comprimento da matriz, mas cria apenas um buraco nela.

Co-funded by the
Erasmus+ Programme
of the European Union

Invocar os Métodos dos Objetos

O método de um objeto pode ser acedido da mesma forma que acedemos às propriedades – usando uma notação de ponto ou aplicando a notação de parenteses retos, como pode ser verificado aqui.

Co-funded by the
Erasmus+ Programme
of the European Union

Manipulação de Valores vs. Referências

Os objetos JS são tipos de referências, o que significa que, quando o programador faz cópias destes, eles estão simplesmente a copiar as referências para esse objeto, enquanto os valores primitivos, como linhas ou números, são atribuídos ou copiados como um valor completo. Este exemplo demonstra esta ideia.

Co-funded by the
Erasmus+ Programme
of the European Union

Manipulação de Valores vs. Referências

- al como pode ser observado, já foi feita uma cópia de uma mensagem variável e foi alterdo o valor dessa mesma cópia. Ambas as variáveis mantêm-se distintas e separadas. No entanto, se este mesmo princípio for aplicado num objeto, o resultado recolhido será diferente.
- Portanto, qualquer alteração feita à variável do utilizador também vai interferir com a variável de pessoa, pois ambas se referem ao mesmo objeto. Concluindo, a mera cópia de um objeto não o duplica, mas copia apenas a referência desse objeto.

Co-funded by the
Erasmus+ Programme
of the European Union

VAMOS PRATICAR!

Vamos abrir o link abaixo para praticar alguns destes conceitos:

https://www.w3resource.com/javascript-exercises/javascript-basic-exercises.php

Co-funded by the
Erasmus+ Programme
of the European Union

code4sp
coding for social promotion

OBRIGADO!

Co-funded by the
Erasmus+ Programme
of the European Union